# BANDITSUM: Extractive Summarization as a Contextual Bandit

**Yue Dong***
Mila/McGill University
yue.dong2@mail.mcgill.ca

**Yikang Shen***
Mila/University of Montréal
yi-kang.shen@umontreal.ca

**Eric Crawford**
Mila/McGill University
eric.crawford@mail.mcgill.ca

**Herke van Hoof**
University of Amsterdam
h.c.vanhoof@uva.nl

**Jackie C.K. Cheung**
Mila/McGill University
jcheung@cs.mcgill.ca

## Abstract

In this work, we propose a novel method for training neural networks to perform single-document extractive summarization without heuristically-generated extractive labels. We call our approach BANDITSUM as it treats extractive summarization as a contextual bandit (CB) problem, where the model receives a document to summarize (the context), and chooses a sequence of sentences to include in the summary (the action). A policy gradient reinforcement learning algorithm is used to train the model to select sequences of sentences that maximize ROUGE score. We perform a series of experiments demonstrating that BANDITSUM is able to achieve ROUGE scores that are better than or comparable to the state-of-the-art for extractive summarization, and converges using significantly fewer update steps than competing approaches. In addition, we show empirically that BANDITSUM performs significantly better than competing approaches when good summary sentences appear late in the source document.

## 1 Introduction

Single-document summarization methods can be divided into two categories: extractive and abstractive. Extractive summarization systems form summaries by selecting and copying text snippets from the document, while abstractive methods aim to generate concise summaries with paraphrasing. This work is primarily concerned with extractive summarization. Though abstractive summarization methods have made strides in recent years, extractive techniques are still very attractive as they are simpler, faster, and more reliably yield semantically and grammatically correct sentences.

Many extractive summarizers work by selecting sentences from the input document (Luhn, 1958; Mihalcea and Tarau, 2004; Wong et al., 2008; Kågebäck et al., 2014; Yin and Pei, 2015; Cao et al., 2015; Yasunaga et al., 2017). Furthermore, a growing trend is to frame this sentence selection process as a sequential binary labeling problem, where binary inclusion/exclusion labels are chosen for sentences one at a time, starting from the beginning of the document, and decisions about later sentences may be conditioned on decisions about earlier sentences. Recurrent neural networks may be trained with stochastic gradient ascent to maximize the likelihood of a set of ground-truth binary label sequences (Cheng and Lapata, 2016; Nallapati et al., 2017). However, this approach has two well-recognized disadvantages. First, it suffers from exposure bias, a form of mismatch between training and testing data distributions which can hurt performance (Ranzato et al., 2015; Bahdanau et al., 2017; Paulus et al., 2018). Second, extractive labels must be generated by a heuristic, as summarization datasets do not generally include ground-truth extractive labels; the ultimate performance of models trained on such labels is thus fundamentally limited by the quality of the heuristic.

An alternative to maximum likelihood training

*Equal contribution.



*Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748
Brussels, Belgium, October 31 - November 4, 2018. 

is to use reinforcement learning to train the model to directly maximize a measure of summary quality, such as the ROUGE score between the generated summary and a ground-truth abstractive summary (Wu and Hu, 2018). This approach has become popular because it avoids exposure bias, and directly optimizes a measure of summary quality. However, it also has a number of downsides. For one, the search space is quite large: for a document of length $T$, there are $2^T$ possible extractive summaries. This makes the exploration problem faced by the reinforcement learning algorithm during training very difficult. Another issue is that due to the sequential nature of selection, the model is inherently biased in favor of selecting earlier sentences over later ones, a phenomenon which we demonstrate empirically in Section 7. The first issue can be resolved to a degree using either a cumbersome maximum likelihood-based pre-training step (using heuristically-generated labels) (Wu and Hu, 2018), or placing a hard upper limit on the number of sentences selected. The second issue is more problematic, as it is inherent to the sequential binary labeling setting.

In the current work, we introduce BANDITSUM, a novel method for training neural network-based extractive summarizers with reinforcement learning. This method does away with the sequential binary labeling setting, instead formulating extractive summarization as a contextual bandit. This move greatly reduces the size of the space that must be explored, removes the need to perform supervised pre-training, and prevents systematically privileging earlier sentences over later ones. Although the strong performance of Lead-3 indicates that good sentences often occur early in the source article, we show in Sections 6 and 7 that the contextual bandit setting greatly improves model performance when good sentences occur late without sacrificing performance when good sentences occur early.

Under this reformulation, BANDITSUM takes the document as input and outputs an *affinity* for each of the sentences therein. An affinity is a real number in $[0, 1]$ which quantifies the model's propensity for including a sentence in the summary. These affinities are then used in a process of repeated sampling-without-replacement which does not privilege earlier sentences over later ones. BANDITSUM is free to process the document as a whole before yielding affinities, which permits affinities for different sentences in the document to depend on one another in arbitrary ways. In our technical section, we show how to apply policy gradient reinforcement learning methods to this setting.

The contributions of our work are as follows:

- We propose a theoretically grounded method, based on the contextual bandit formalism, for training neural network-based extractive summarizers with reinforcement learning. Based on this training method, we propose the BANDITSUM system for extractive summarization.
- We perform experiments demonstrating that BANDITSUM obtains state-of-the-art performance on a number of datasets and requires significantly fewer update steps than competing approaches.
- We perform human evaluations showing that in the eyes of human judges, summaries created by BANDITSUM are less redundant and of higher overall quality than summaries created by competing approaches.
- We provide evidence, in the form of experiments in which models are trained on subsets of the data, that the improved performance of BANDITSUM over competitors stems in part from better handling of summary-worthy sentences that come near the end of the document (see Section 7).

## 2 Related Work

Extractive summarization has been widely studied in the past. Recently, neural network-based methods have been gaining popularity over classical methods (Luhn, 1958; Gong and Liu, 2001; Conroy and O'leary, 2001; Mihalcea and Tarau, 2004; Wong et al., 2008), as they have demonstrated stronger performance on large corpora. Central to the neural network-based models is the encoder-decoder structure. These models typically use either a convolution neural network (Kalchbrenner et al., 2014; Kim, 2014; Yin and Pei, 2015; Cao et al., 2015), a recurrent neural network (Chung et al., 2014; Cheng and Lapata, 2016; Nallapati et al., 2017), or a combination of the two (Narayan et al., 2018; Wu and Hu, 2018) to create sentence and document representations, using word embeddings (Mikolov et al., 2013; Pennington et al.,

2014) to represent words at the input level. These vectors are then fed into a decoder network to generate the output summary.

The use of reinforcement learning (RL) in extractive summarization was first explored by Ryang and Abekawa (2012), who proposed to use the TD($\lambda$) algorithm to learn a value function for sentence selection. Rioux et al. (2014) improved this framework by replacing the learning agent with another TD($\lambda$) algorithm. However, the performance of their methods was limited by the use of shallow function approximators, which required performing a fresh round of reinforcement learning for every new document to be summarized. The more recent work of Paulus et al. (2018) and Wu and Hu (2018) use reinforcement learning in a sequential labeling setting to train abstractive and extractive summarizers, respectively, while Chen and Bansal (2018) combines both approaches, applying abstractive summarization to a set of sentences extracted by a pointer network (Vinyals et al., 2015) trained via REINFORCE. However, pre-training with a maximum likelihood objective is required in all of these models.

The two works most similar to ours are Yao et al. (2018) and Narayan et al. (2018). Yao et al. (2018) recently proposed an extractive summarization approach based on deep Q learning, a type of reinforcement learning. However, their approach is extremely computationally intensive (a minimum of 10 days before convergence), and was unable to achieve ROUGE scores better than the best maximum likelihood-based approach. Narayan et al. (2018) uses a cascade of filters in order to arrive at a set of candidate extractive summaries, which we can regard as an approximation of the true action space. They then use an approximation of a policy gradient method to train their neural network to select summaries from this approximated action space. In contrast, BANDITSUM samples directly from the true action space, and uses exact policy gradient parameter updates.

## 3 Extractive Summarization as a Contextual Bandit

Our approach formulates extractive summarization as a contextual bandit which we then train an agent to solve using policy gradient reinforcement learning. A bandit is a decision-making formalization in which an agent repeatedly chooses one of several actions, and receives a reward based on this choice. The agent's goal is to quickly learn which action yields the most favorable distribution over rewards, and choose that action as often as possible. In a *contextual* bandit, at each trial, a context is sampled and shown to the agent, after which the agent selects an action and receives a reward; importantly, the rewards yielded by the actions may depend on the sampled context. The agent must quickly learn which actions are favorable in which contexts. Contextual bandits are a subset of Markov Decision Processes in which every episode has length one.

Extractive summarization may be regarded as a contextual bandit as follows. Each document is a context, and each ordered subset of a document's sentences is a different action. Formally, assume that each context is a document $d$ consisting of sentences $s = (s_1, \ldots, s_{N_d})$, and that each action is a length-$M$ sequence of unique sentence indices $i = (i_1, \ldots, i_M)$ where $i_t \in \{1, \ldots, N_d\}$, $i_t \neq i_{t'}$ for $t \neq t'$, and $M$ is an integer hyper-parameter. For each $i$, the extractive summary induced by $i$ is given by $(s_{i_1}, \ldots, s_{i_M})$. An action $i$ taken in context $d$ is given a reward $R(i, a)$, where $a$ is the gold-standard abstractive summary that is paired with document $d$, and $R$ is a scalar reward function quantifying the degree of match between $a$ and the summary induced by $i$.

A policy for extractive summarization is a neural network $p_\theta(\cdot|d)$, parameterized by a vector $\theta$, which, for each input document $d$, yields a probability distribution over index sequences. Our goal is to find parameters $\theta$ which cause $p_\theta(\cdot|d)$ to assign high probability to index sequences that induce extractive summaries that a human reader would judge to be of high-quality. We achieve this by maximizing the following objective function with respect to parameters $\theta$:

$$J(\theta) = E\left[R(i, a)\right] \tag{1}$$

where the expectation is taken over documents $d$ paired with gold-standard abstractive summaries $a$, as well as over index sequences $i$ generated according to $p_\theta(\cdot|d)$.

### 3.1 Policy Gradient Reinforcement Learning

Ideally, we would like to maximize (1) using gradient ascent. However, the required gradient cannot be obtained using usual techniques (e.g. simple backpropagation) because $i$ must be discretely sampled in order to compute $R(i, a)$.

Fortunately, we can use the likelihood ratio gradient estimator from reinforcement learning and stochastic optimization (Williams, 1992; Sutton et al., 2000), which tells us that the gradient of this function can be computed as:

$$\nabla_\theta J(\theta) = E\left[\nabla_\theta \log p_\theta(i|d) R(i,a)\right] \quad (2)$$

where the expectation is taken over the same variables as (1).

Since we typically do not know the exact document distribution and thus cannot evaluate the expected value in (2), we instead estimate it by sampling. We found that we obtained the best performance when, for each update, we first sample one document/summary pair $(d, a)$, then sample $B$ index sequences $i^1, \ldots, i^B$ from $p_\theta(\cdot|d)$, and finally take the empirical average:

$$\nabla_\theta J(\theta) \approx \frac{1}{B} \sum_{b=1}^{B} \nabla_\theta \log p_\theta(i^b|d) R(i^b, a) \quad (3)$$

This overall learning algorithm can be regarded as an instance of the REINFORCE policy gradient algorithm (Williams, 1992).

### 3.2 Structure of $p_\theta(\cdot|d)$

There are many possible choices for the structure of $p_\theta(\cdot|d)$; we opt for one that avoids privileging early sentences over later ones. We first decompose $p_\theta(\cdot|d)$ into two parts: $\pi_\theta$, a deterministic function which contains all the network's parameters, and $\mu$, a probability distribution parameterized by the output of $\pi_\theta$. Concretely:

$$p_\theta(\cdot|d) = \mu(\cdot|\pi_\theta(d)) \quad (4)$$

Given an input document $d$, $\pi_\theta$ outputs a real-valued vector of *sentence affinities* whose length is equal to the number of sentences in the document (i.e. $\pi_\theta(d) \in \mathbb{R}^{N_d}$) and whose elements fall in the range $[0, 1]$. The $t$-th entry $\pi(d)_t$ may be roughly interpreted as the network's propensity to include sentence $s_t$ in the summary of $d$.

Given sentence affinities $\pi_\theta(d)$, $\mu$ implements a process of repeated sampling-without-replacement. This proceeds by repeatedly normalizing the set of affinities corresponding to sentences that have not yet been selected, thereby obtaining a probability distribution over unselected sentences, and sampling from that distribution to obtain a new sentence to include. This normalize-and-sample step is repeated $M$ times, yielding $M$ unique sentences to include in the summary.

At each step of sampling-without-replacement, we also include a small probability $\epsilon$ of sampling uniformly from all remaining sentences. This is used to achieve adequate exploration during training, and is similar to the $\epsilon$-greedy technique from reinforcement learning.

Under this sampling scheme, we have the following expression for $p_\theta(i|d)$:

$$\prod_{j=1}^{M} \left( \frac{\epsilon}{N_d - j + 1} + \frac{(1-\epsilon)\pi(d)_{i_j}}{z(d) - \sum_{k=1}^{j-1} \pi(d)_{i_k}} \right) \quad (5)$$

where $z(d) = \sum_t \pi(d)_t$. For index sequences that have length different from $M$, or that contain duplicate indices, we have $p_\theta(i|d) = 0$. Using this expression, it is straightforward to use automatic differentiation software to compute $\nabla_\theta \log p_\theta(i|d)$, which is required for the gradient estimate in (3).

### 3.3 Baseline for Variance Reduction

Our sample-based gradient estimate can have high variance, which can slow the learning. One potential cause of this high variance can be seen by inspecting (3), and noting that it basically acts to change the probability of a sampled index sequence to an extent determined by the reward $R(i, a)$. However, since ROUGE scores are always positive, the probability of every sampled index sequence is increased, whereas intuitively, we would prefer to decrease the probability of sequences that receive a comparatively low reward, even if it is positive. This can be remedied by the introduction of a so-called baseline which is subtracted from all rewards.

Using a baseline $\overline{r}$, our sample-based estimate of $\nabla_\theta J(\theta)$ becomes:

$$\frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \log p_\theta(i^b|d)(R(i^b, a) - \overline{r}) \quad (6)$$

It can be shown that the introduction of $\overline{r}$ does not bias the gradient estimator and can significantly reduce its variance if chosen appropriately (Sutton et al., 2000).

There are several possibilities for the baseline, including the long-term average reward and the average reward across different samples for one document-summary pair. We choose an approach known as self-critical reinforcement learning, in which the test-time performance of the current model is used as the baseline (Ranzato et al., 2015;

Rennie et al., 2017; Paulus et al., 2018). More concretely, after sampling the document-summary pair $(d, a)$, we greedily generate an index sequence using the current parameters $\theta$:

$$i_{greedy} = \arg\max_{i} p_\theta(i|d) \quad (7)$$

and calculate the baseline for the current update as $\overline{r} = R(i_{greedy}, a)$. This baseline has the intuitively satisfying property of only increasing the probability of a sampled label sequence when the summary it induces is better than what would be obtained by greedy decoding.

### 3.4 Reward Function

A final consideration is a concrete choice for the reward function $R(i, a)$. Throughout this work we use:

$$R(i,a) = \frac{1}{3}(\text{ROUGE-1}_f(i,a) + \text{ROUGE-2}_f(i,a) + \text{ROUGE-L}_f(i,a)). \quad (8)$$

The above reward function optimizes the average of all the ROUGE variants (Lin, 2004) while balancing precision and recall.

## 4 Model

In this section, we discuss the concrete instantiations of the neural network $\pi_\theta$ that we use in our experiments. We break $\pi_\theta$ up into two components: a document encoder $f_{\theta 1}$, which outputs a sequence of sentence feature vectors $(h_1, \ldots, h_{N_d})$ and a decoder $g_{\theta 2}$ which yields sentence affinities:

$$h_1, \ldots, h_{N_d} = f_{\theta 1}(d) \quad (9)$$

$$\pi_\theta(d) = g_{\theta 2}(h_1, \ldots, h_{N_d}) \quad (10)$$

**Encoder.** Features for each sentence in isolation are first obtained by applying a word-level Bidirectional Recurrent Neural Network (BiRNN) to the embeddings for the words in the sentence, and averaging the hidden states over words. A separate sentence-level BiRNN is then used to obtain a representations $h_i$ for each sentence in the context of the document.

**Decoder.** A multi-layer perceptron is used to map from the representation $h_t$ of each sentence through a final sigmoid unit to yield sentence affinities $\pi_\theta(d)$.

The use of a bidirectional recurrent network in the encoder is crucial, as it allows the network to process the document as a whole, yielding representations for each sentence that take all other sentences into account. This procedure is necessary to deal with some aspects of summary quality such as redundancy (avoiding the inclusion of multiple sentences with similar meaning), which requires the affinities for different sentences to depend on one another. For example, to avoid redundancy, if the affinity for some sentence is high, then sentences which express similar meaning should have low affinities.

## 5 Experiments

In this section, we discuss the setup of our experiments. We first discuss the corpora that we used and our evaluation methodology. We then discuss the baseline methods against which we compared, and conclude with a detailed overview of the settings of the model parameters.

### 5.1 Corpora

Three datasets are used for our experiments: the CNN, the Daily Mail, and combined CNN/Daily Mail (Hermann et al., 2015; Nallapati et al., 2016). We use the standard split of Hermann et al. (2015) for training, validating, and testing and the same setting without *anonymization* on the three corpus as See et al. (2017). The Daily Mail corpus has 196,557 training documents, 12,147 validation documents and 10,397 test documents; while the CNN corpus has 90,266/1,220/1,093 documents, respectively.

### 5.2 Evaluation

The models are evaluated based on ROUGE (Lin, 2004). We obtain our ROUGE scores using the standard pyrouge package[1] for the test set evaluation and a faster python implementation of the ROUGE metric[2] for training and evaluating on the validation set. We report the F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L, which compute the uniform, bigram, and longest common subsequence overlapping with the reference summaries.

### 5.3 Baselines

We compare BANDITSUM with other extractive methods including: the Lead-3 model, SummaRuNNer (Nallapati et al., 2017), Refresh

[1] https://pypi.python.org/pypi/pyrouge/0.1.3

[2] We use the modified version based on https://github.com/pltrdy/rouge

(Narayan et al., 2018), RNES (Wu and Hu, 2018), DQN (Yao et al., 2018), and NN-SE (Cheng and Lapata, 2016). The Lead-3 model simply produces the leading three sentences of the document as the summary.

### 5.4 Model Settings

We use 100-dimensional Glove embeddings (Pennington et al., 2014) as our embedding initialization. We do not limit the sentence length, nor the maximum number of sentences per document. We use one-layer BiLSTM for word-level RNN, and two-layers BiLSTM for sentence-level RNN. The hidden state dimension is 200 for each direction on all LSTMs. For the decoder, we use a feed-forward network with one hidden layer of dimension 100.

During training, we use Adam (Kingma and Ba, 2015) as the optimizer with the learning rate of $5e^{-5}$, beta parameters $(0, 0.999)$, and a weight decay of $1e^{-6}$, to maximize the objective function defined in equation (1). We employ gradient clipping of 1 to regularize our model. At each iteration, we sample $B = 20$ times to estimate the gradient defined in equation 3. For our system, the reported performance is obtained within two epochs of training [3].

At the test time, we pick sentences sorted by the predicted probabilities until the length limit is reached. The full-length ROUGE F1 score is used as the evaluation metric. For $M$, the number of sentences selected per summary, we use a value of 3, based on our validation results as well as on the settings described in Nallapati et al. (2017).

## 6 Experiment Results

In this section, we present quantitative results from the ROUGE evaluation and qualitative results based on human evaluation. In addition, we demonstrate the stability of our RL model by comparing the validation curve of BANDITSUM with SummaRuNNer (Nallapati et al., 2017) trained with a maximum likelihood objective.

### 6.1 Rouge Evaluation

We present the results of comparing BANDITSUM to several baseline algorithms[4] on the CNN/Daily Mail corpus in Tables 1 and 2. Compared to other extractive summarization systems, BANDITSUM achieves performance that is significantly better than two RL-based approaches, Refresh (Narayan et al., 2018) and DQN (Yao et al., 2018), as well as SummaRuNNer, the state-of-the-art maximum liklihood-based extractive summarizer (Nallapati et al., 2017). BANDITSUM performs a little better than RNES (Wu and Hu, 2018) in terms of ROUGE-1 and slightly worse in terms of ROUGE-2. However, RNES requires pre-training with the maximum likelihood objective on heuristically-generated extractive labels; in contrast, BANDITSUM is very light-weight and converges significantly faster. We discuss the advantage of framing the extractive summarization based on the contextual bandit (BANDITSUM) over the sequential binary labeling setting (RNES) in the discussion Section 7.

We also noticed that different choices for the policy gradient baseline (see Section 3.3) in BANDITSUM affect learning speed, but do not significantly affect asymptotic performance. Models trained with an average reward baseline learned most quickly, while models trained with three different baselines (greedy, average reward in a

| Model | ROUGE | | |
|---|---|---|---|
| | 1 | 2 | L |
| Lead(Narayan et al., 2018) | 39.6 | 17.7 | 36.2 |
| Lead-3(ours) | 40.0 | 17.5 | 36.2 |
| SummaRuNNer | 39.6 | 16.2 | 35.3 |
| DQN | 39.4 | 16.1 | 35.6 |
| Refresh | 40.0 | 18.2 | 36.6 |
| RNES w/o coherence | 41.3 | **18.9** | **37.6** |
| BANDITSUM | **41.5** | 18.7 | **37.6** |

Table 1: Performance comparison of different extractive summarization models on the combined CNN/Daily Mail test set using full-length F1.

| Model | CNN | | | Daily Mail | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | L | 1 | 2 | L |
| Lead-3 | 28.8 | 11.0 | 25.5 | 41.2 | 18.2 | 37.3 |
| NN-SE | 28.4 | 10.0 | 25.0 | 36.2 | 15.2 | 32.9 |
| Refresh | 30.4 | **11.7** | 26.9 | 41.0 | 18.8 | 37.7 |
| BANDITSUM | **30.7** | 11.6 | **27.4** | **42.1** | **18.9** | **38.3** |

Table 2: The full-length ROUGE F1 scores of various extractive models on the CNN and the Daily Mail test set separately.

[3] Our code can be found at https://github.com/yuedongP/summarization_RL

[4] Due to different pre-processing methods and different numbers of selected sentences, several papers report different Lead scores (Narayan et al., 2018; See et al., 2017). We use the test set provided by Narayan et al. (2018). Since their Lead score is a combination of Lead-3 for CNN and Lead-4 for Daily Mail, we recompute the Lead-3 scores for both CNN and Daily Mail with the preprocessing steps used in See et al. (2017). Additionally, our results are not directly comparable to results based on the anonymized dataset used by Nallapati et al. (2017).

batch, average global reward) all perform roughly the same after training for one epoch. Models trained without a baseline were found to underperform other baseline choices by about 2 points of ROUGE score on average.

### 6.2 Human Evaluation

We also conduct a qualitative evaluation to understand the effects of the improvements introduced in BANDITSUM on human judgments of the generated summaries. To assess the effect of training with RL rather than maximum likelihood, in the first set of human evaluations we compare BANDITSUM with the state-of-the-art maximum likelihood-based model SummaRuNNer. To evaluate the importance of using an exact, rather than approximate, policy gradient to optimize ROUGE scores, we perform another human evaluation comparing BANDITSUM and Refresh, an RL-based method that uses the an approximation of the policy gradient.

We follow a human evaluation protocol similar to the one used in Wu and Hu (2018). Given a set of $N$ documents, we ask $K$ volunteers to evaluate the summaries extracted by both systems. For each document, a reference summary, and a pair of randomly ordered extractive summaries (one generated by each of the two models) is presented to the volunteers. They are asked to compare and rank the extracted summaries along three dimensions: overall, coverage, and non-redundancy.

| Model | Overall | Coverage | Non-Redundancy |
|---|---|---|---|
| SummaRuNNer | 1.67 | **1.46** | 1.70 |
| BANDITSUM | **1.33** | 1.54 | **1.30** |

Table 3: Average rank of human evaluation based on 5 participants who expressed 57 pairwise preferences between the summaries generated by SummaRuNNer and BANDITSUM. The model with the lower score is better.

| Model | Overall | Coverage | Non-Redundancy |
|---|---|---|---|
| Refresh | 1.53 | **1.34** | 1.55 |
| BANDITSUM | **1.50** | 1.58 | **1.30** |

Table 4: Average rank of manual evaluation with 4 participants who expressed 20 pairwise preferences between the summaries generated by Refresh and our system. The model with the lower score is better.

To compare with SummaRuNNer, we randomly sample 57 documents from the test set of DailyMail and ask 5 volunteers to evaluate the extracted summaries. While comparing with Refresh, we use the 20 documents (10 CNN and 10 DailyMail) provided by Narayan et al. (2018) to 4 volunteers. Tables 3 and 4 show the results of human evaluation in these two settings. BANDITSUM is shown to be better than Refresh and SummaRuNNer in terms of overall quality and non-redundancy. These results indicate that the use of the true policy gradient, rather than the approximation used by Refresh, improves overall quality. It is interesting to observe that, even though BANDITSUM does not have an explicit redundancy avoidance mechanism, it actually outperforms the other systems on non-redundancy.

### 6.3 Learning Curve

Reinforcement learning methods are known for sometimes being unstable during training. However, this seems to be less of a problem for BANDITSUM, perhaps because it is formulated as a contextual bandit rather than a sequential labeling problem. We show this by comparing the validation curves generated by BANDITSUM and the state-of-the-art maximum likelihood-based model – SummaRuNNer (Nallapati et al., 2017) (Figure 1).

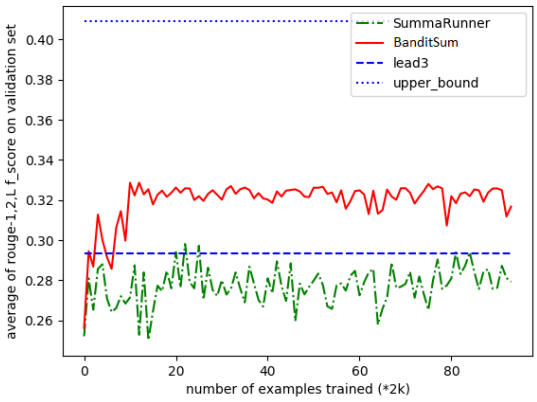


Figure 1: Average of ROUGE-1,2,L F1 scores on the Daily Mail validation set within one epoch of training on the Daily Mail training set. The x-axis (multiply by 2,000) indicates the number of data example the algorithms have seen. The supervised labels in SummaRuNNer are used to estimate the `upper_bound`.

From Figure 1, we observe that BANDITSUM converges significantly more quickly to good results than SummaRuNNer. Moreover, there is less variance in the performance of BANDITSUM.

One possible reason is that extractive summarization does not have well-defined supervised labels. There exists a mismatch between the provided labels and human-generated abstractive summaries. Hence, the gradient, computed from the maximum likelihood loss function, is not optimizing the evaluation metric of interest. Another important message is that both models are still far from the estimated upper bound[5], which shows that there is still significant room for improvement.

### 6.4 Run Time

On CNN/Daily mail dataset, our model's time-per-epoch is about 25.5 hours on a TITAN Xp. We trained the model for 3 epochs, which took about 76 hours in total. For comparison, DQN took about 10 days to train on a GTX 1080 (Yao et al., 2018). Refresh took about 12 hours on a single GPU to train (Narayan et al., 2018). Note that this figure does not take into account the significant time required by Refresh for pre-computing ROUGE scores.

## 7 Discussion: Contextual Bandit Setting Vs. Sequential Full RL Labeling

We conjecture that the contextual bandit (CB) setting is a more suitable framework for modeling extractive summarization than the sequential binary labeling setting, especially in the cases when good summary sentences appear later in the document. The intuition behind this is that models based on the sequential labeling setting are affected by the order of the decisions, which biases towards selecting sentences that appear earlier in the document. By contrast, our CB-based RL model has more flexibility and freedom to explore the search space, as it samples the sentences without replacement based on the affinity scores. Note that although we do not explicitly make the selection decisions in a sequential fashion, the sequential information about dependencies between sentences is implicitly embedded in the affinity scores, which are produced by bi-directional RNNs.

We provide empirical evidence for this conjecture by comparing BANDITSUM to the sequential RL model proposed by Wu and Hu (2018) (Figure 2) on two subsets of the data: one with good summary sentences appearing early in the article, while the other contains articles where good summary sentences appear late. Specifically, we construct two evaluation datasets by selecting the first 50 documents ($D_{early}$, i.e., best summary occurs early) and the last 50 documents ($D_{late}$, i.e., best summary occurs late) from a sample of 1000 documents that is ordered by the average extractive label index $\overline{idx}$. Given an article with $n$ sentences indexed from $1, \ldots, n$ and a greedy extractive labels set with three sentences $(i, j, k)$[6], the average index for the extractive label is computed by $\overline{idx} = (i + j + k)/3n$.

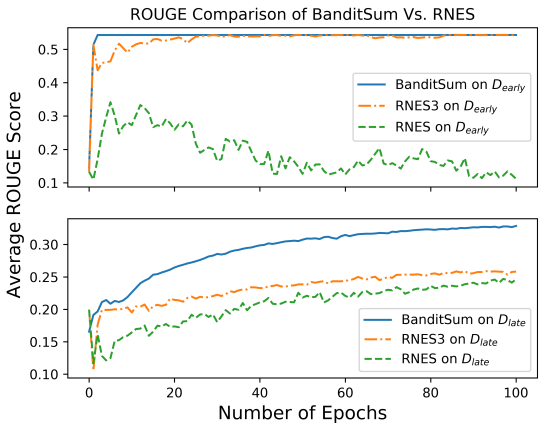


Figure 2: Model comparisons of the average value for ROUGE-1,2,L F1 scores ($\overline{f}$) on $D_{early}$ and $D_{late}$. For each model, the results were obtained by averaging $\overline{f}$ across ten trials with 100 epochs in each trail. $D_{early}$ and $D_{late}$ consist of 50 articles each, such that the good summary sentences appear early and late in the article, respectively. We observe a significant advantage of BANDITSUM compared to RNES and RNES3 (based on the sequential binary labeling setting) on $D_{late}$.

Given these two subsets of the data, three different models (BANDITSUM, RNES and RNES3) are trained and evaluated on each of the two datasets without extractive labels. Since the original sequential RL model (RNES) is unstable without supervised pre-training, we propose the RNES3 model that is limited to select no more then three sentences. Starting with random initializations without supervised pre-training, we train each model ten times for 100 epochs and plot the learning curve of the average ROUGE-F1 score computed based on the trained model in Figure 2. We can clearly see that BANDITSUM finds a better so-

[5]The supervised labels for the `upper_bound` estimation are obtained using the heuristic described in Nallapati et al. (2017).

[6]For each document, a length-3 extractive summary with near-optimal ROUGE score is selected following the heuristic proposed by Nallapati et al. (2017).

lution more quickly than RNES and RNES3 on both datasets. Moreover, it displays a significantly speed-up in the exploration and finds the best solution when good summary sentences appeared later in the document ($D_{late}$).

## 8 Conclusion

In this work, we presented a contextual bandit learning framework, BANDITSUM , for extractive summarization, based on neural networks and reinforcement learning algorithms. BANDITSUM does not require sentence-level extractive labels and optimizes ROUGE scores between summaries generated by the model and abstractive reference summaries. Empirical results show that our method performs better than or comparable to state-of-the-art extractive summarization models which must be pre-trained on extractive labels, and converges using significantly fewer update steps than competing approaches. In future work, we will explore the direction of adding an extra coherence reward (Wu and Hu, 2018) to improve the quality of extracted summaries in terms of sentence discourse relation.

## Acknowledgements

The research was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC). The authors would like to thank Compute Canada for providing the computational resources.

## References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*.

John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 406–407. ACM.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 19–25. ACM.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. In *Workshop on Continuous Vector Space Models and their Compositionality (CVSC) at EACL*, pages 31–39.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference for Learning Representations (ICLR)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR) Workshop*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gùlçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7008–7024.

Cody Rioux, Sadid A Hasan, and Yllias Chali. 2014. Fear the reaper: A system for automatic multi-document summarization with reinforcement learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 681–690.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*, pages 256–265.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1073–1083.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1057–1063.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2692–2700.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 985–992.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Kaichun Yao, Libo Zhang, Tiejian Luo, and Yanjun Wu. 2018. Deep reinforcement learning for extractive document summarization. *Neurocomputing*, 284:52–62.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL)*, pages 452–462.

Wenpeng Yin and Yulong Pei. 2015. Optimizing sentence modeling and selection for document summarization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1383–1389.